Efficient Frequency Domain Filtering Algorithm for Small Data Sets

Canadian Technical Report of
Fisheries and Aquatic Sciences
No. 1559

Fisheries and Oceans Pêches et Océans

Canada

88-01216/

Canadian Technical Report of Fisheries and Aquatic Sciences

Technical reports contain scientific and technical information that contributes to existing knowledge but which is not normally appropriate for primary literature. Technical reports are directed primarily toward a worldwide audience and have an international distribution. No restriction is placed on subject matter and the series reflects the broad interests and policies of the Department of Fisheries and Oceans, namely, fisheries and aquatic sciences.

Technical reports may be cited as full publications. The correct citation appears above the abstract of each report. Each report is abstracted in *Aquatic Sciences and Fisheries Abstracts* and indexed in the Department's annual index to scientific and technical publications.

Numbers 1 456 in this series were issued as Technical Reports of the Fisheries Research Board of Canada. Numbers 457 714 were issued as Department of the Environment, Fisheries and Marine Service, Research and Development Directorate Technical Reports. Numbers 715 924 were issued as Department of Fisheries and the Environment, Fisheries and Marine Service Technical Reports. The current series name was changed with report number 925.

Technical reports are produced regionally but are numbered nationally. Requests for individual reports will be filled by the issuing establishment listed on the front cover and title page. Out-of-stock reports will be supplied for a fee by commercial agents.

Rapport technique canadien des sciences halieutiques et aquatiques

Les rapports techniques contiennent des renseignements scientifiques et techniques qui constituent une contribution aux connaissances actuelles, mais qui ne sont pas normalement appropriés pour la publication dans un journal scientifique. Les rapports techniques sont destinés essentiellement à un public international et ils sont distribués à cet échelon. Il n'y a aucune restriction quant au sujet; de fait, la série reflète la vaste gamme des intérêts et des politiques du ministère des Pêches et des Océans, c'est-à-dire les sciences halieutiques et aquatiques.

Les rapports techniques peuvent être cités comme des publications complètes. Le titre exact paraît au-dessus du résumé de chaque rapport. Les rapports techniques sont résumés dans la revue *Résumés des sciences aquatiques et halieutiques*, et ils sont classés dans l'index annual des publications scientifiques et techniques du Ministère.

Les numéros 1 à 456 de cette série ont été publiés à titre de rapports techniques de l'Office des recherches sur les pêcheries du Canada. Les numéros 457 à 714 sont parus à titre de rapports techniques de la Direction générale de la recherche et du développement, Service des pêches et de la mer, ministère de l'Environnement. Les numéros 715 à 924 ont été publiés à titre de rapports techniques du Service des pêches et de la mer, ministère des Pêches et de l'Environnement. Le nom actuel de la série a été établi lors de la parution du numéro 925.

Les rapports techniques sont produits à l'échelon régional, mais numérotés à l'échelon national. Les demandes de rapports seront satisfaites par l'établissement auteur dont le nom figure sur la couverture et la page du titre. Les rapports épuisés seront fournis contre rétribution par des agents commerciaux.

Canadian Technical Report of

Fisheries and Aquatic Sciences No. 1559

May 1987

EFFICIENT FREQUENCY DOMAIN FILTERING ALGORITHM FOR SMALL DATA SETS

by

D. W. Welch

Department of Fisheries and Oceans

Fisheries Research Branch

Pacific Biological Station

Nanaimo, British Columbia, V9R 5K6

(C) Ministry of Supply and Services Canada 1987

Cat. No. Fs 97-6/1559E ISSN 0706-6457

Correct citation for this publication:

Welch, D. W. 1987 Efficient frequency domain filtering algorithm for small data sets. Can. Tech. Rep. Fish. Aquat. Sci. 1559: 27 p.

TABLE OF CONTENTS

	Page
Abstract/Résumé	iv
Introduction	1
Use of the filtering codes	2
Algorithm flow	3
Pascal version	4
Fortran version	5
Test programs	6
Summary	14
Acknowledgments	15
Literature cited	16
Appendix I. Frequency domain filtering algorithm: Pascal	17
Appendix II. Frequency domain filtering algorithm: Fortran	22

NOTICE OF LIABILITY:

The Government of Canada shall have no liability or responsibility to any person or entity with respect to any liability, loss or damage caused directly or indirectly by this software, including but not limited to any interuption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of this software.

There is no service associated with this software and any defect or fault found therein will not be the responsibility of the Government of Canada.

ABSTRACT

Welch, D. W. 1987 Efficient frequency domain filtering algorithm for small data sets. Can. Tech. Rep. Fish. Aquat. Sci. 1559: 27p

This report provides ANSI-1977 Pascal and Fortran source codes suitable for filtering time series of biological data using criteria determined by the age-structure of the population. The codes allow fast and accurate filtering of relatively short time series ($\leq$ 100 points) using a Fourier transform method that circumvents the usual requirement that the time series length be an integer power of two. The codes are designed to be used as subroutines from within the user's own programs. Several test programs are included to illustrate the use of the codes.

RÉSUMÉ

Welch, D. W. 1987 Efficient frequency domain filtering algorithm for small data sets. Can. Tech. Rep. Fish. Aquat. Sci. 1559: 27p

Sont présentés les codes sources en Fortran et en Pascal (ANSI-1977) que l'on peut utiliser pour le filtrage des séries chronologiques de données biologiques en appliquant des critères déterminés par la structure par âge de la population. Ces codes permettent un filtrage rapide et exact des séries chronologiques relativement courtes (100 points et moins) au moyen d'une méthode faisant appel à la transformée de Fourier; avec cette méthode, il n'est pas nécessaire que la longueur de la série chronologique soit de deux élevé à une puissance entière. Ces codes sont destinés à être utilisés comme sous-programmes. Plusiers programmes permettant de voir l'utilisation des codes sont présentés.

INTRODUCTION

This report provides Pascal and Fortran versions of the frequency-domain filtering algorithm described in Welch (1987a). These algorithms have been written as user-callable routines to allow maximum flexibility in their use by individuals interested in applying filtering theory to their own analyses. Machine readable copies of the source code (either floppy disk or nine track tape) are available from the author.

The basic theory behind age-structure based filtering is developed in Welch (1986) using the example of removing environmental noise from recruitment data. However, the uses of filtering theory in the study of population dynamics are potentially much wider. Examples include the filtering of time series of fecundity or maturity data to better clarify any possible dependence on population density, or the examination of the difference between filtered and unfiltered data for evidence of correlation with specific environmental factors.

Two specific examples where age-structure based filtering of recruitment data is applicable to the analysis of density-dependent regulation occur when survival to recruitment is thought to depend on either the initial density of the 0-group or on the density of a cannibalistic adult population. In either case the need is to filter a specified time series to remove the variability caused by other factors (environmental or measurement error, for example), and to thereby enhance the relative strength of the density-dependent signal.

The frequency domain filtering algorithm which the present codes document is used in Welch (1987a) to demonstrate the increased precision of stock-recruitment parameter estimates made possible by filtering recruitment data prior to regression analysis. This paper also shows that in some situations filtering can improve the accuracy of these parameter estimates as well. The reader should refer to Welch (1987a) for a more complete description of the Monte Carlo tests and the conditions under which these conclusions apply.

Two other documents may also be of use in understanding the rationale for applying filtering theory to questions of population regulation in iteroparous (age-structured) populations. First, the letter by Evans (1987) and the reply by Welch (1987b), may help to clarify questions about the information content of stock-recruitment data. Second, analysis of three simple age-structured models using filtering theory (Welch 1987c) should help to establish in the reader's mind exactly how filtering can (first) identify and remove nondensity-dependent variability from a time series and (second) the need for care in correctly specifying the age-structure. The final example included in the present document is in fact the code necessary to reproduce the numerical example given in Welch (1987c).

USE OF THE FILTERING CODES

Appendices I & II provide the source codes for the frequency domain filtering algorithm in, respectively, the Pascal and Fortran languages. The original development of the algorithm and its testing were done in Pascal on a microcomputer using Borland International's TurboPascal(c) compiler[1]. Testing of the Fortran version was done on a microcomputer using the WATCOM Products Inc. WATFOR-77(c) compiler[2]. Because TurboPascal tends to produce much faster and more compact code, as well as being extremely easy to use, the Fortran translation has been provided primarily for the convenience of individuals wishing to use Fortran on a mainframe computer.

Both Pascal and Fortran versions meet ANSI-1977 standards, so there should be little difficulty in using the codes on different computers. Neither the size of these codes nor their execution times should prevent their routine use on small microcomputers. However, I recommend the use of the Intel 8087 series math coprocessor when using a microcomputer because of the large amount of real number multiplication necessary, but even here the time required to filter most time series encountered in fisheries work is small (< 1 min.). The use of a mathematics coprocessor speeds up the execution of the filtering code by a factor of approximately 25.

The set-up of the program using the filtering procedure is designed to be as simple as possible. The time series to be filtered is entered into the array X in chronological sequence, and its length specified in the variable LenX. The age-structure coefficients defining how the sum of past recruitments determines the population density to be compared with the recruitment time series ("stock") is then entered into the array RED, and the length specified in LenRED. The filtering algorithm is then invoked, and the filtered output is returned in FiltX. The user is free to substitute any properly declared names for these variables.

An important difference between the specification of the vector to be filtered, X, and the vector of coefficients determining the filtering criteria should be noted. The time series X should contain only the data to be filtered; it must not be extended in any way. In contrast, it is often desirable to increase the length of the age-structure coefficients by padding the array. This may be done in one of two ways: either by padding with zeros for ages known not to contribute to forming the stock (for example, sexually immature or senescent age-groups), or by using plausible assumptions to extrapolate the known age-structure coefficients to ages for which little or no data exists. For example, a reasonable assumption in some cases might be to extrapolate the reproductive effort at very old ages by assuming the mortality rate estimated for intermediate age fish holds, and by estimating fecundity and maturity rates as allometric functions of body size.

---

1. Borland International, 4113 Scotts Valley Dr., Scotts Valley, Calif., U.S.A. 95066.
2. WATCOM Products Inc., 415 Phillip St., Waterloo, Ontario, Canada N2L 3X2

The reason for extrapolating the age-structure lies in the need to correctly calculate the frequency-dependent attenuation of the recruitment variance. This is done by multiplying the Fourier coefficients of the recruitment time series by the gain of the Fourier transformed age-structure (Welch 1987a). However, the frequencies at which these quantities are calculated will not correspond unless the lengths of the input time series and the age-structure are exact integer multiples. In such cases it is necessary to interpolate the gain at the Fourier frequencies for the recruitment data.

Interpolation is obviously most accurate if the frequency separation of the gain estimates is small. The number of frequencies at which the gain can be calculated is equal to half the time series length plus one (assuming an odd time series length), so the frequency separation is reduced as the age-structure is extended. Whether the RED should be extended with zeros or with extrapolated estimates of the age-structure is a matter for the user's judgement. Note, however, that interpolation is unnecessary when the length of the age-structure is an integer multiple of the time series length, since in this case the Fourier frequencies at which the two sets of values are calculated will coincide.

ALGORITHM FLOW

The flow of both Pascal and Fortran versions of the filtering code are identical, and will be discussed here. The implementation of each version of the code is considered separately below because of the different declarations needed in the calling programs.

Filtering proceeds as follows. First, the gain corresponding to the age-structure is calculated using Goertzel's (1960) recursive method for determining the Fourier coefficents. This procedure is used to speed up calculation of the Fourier transform while sidestepping restrictions on the length of the time series required by fast transform methods.

The actual filtering of the input time series is then begun. The linear time trend is removed to reduce the possibility of introducing high frequency artifacts into the data by Fourier transformation. The data is then Fourier transformed to its frequency domain form, which is a frequency specific set of complex-valued coefficients. In this form the variance in the input time series is simply attenuated by multiplying the Fourier coefficients by the gain of the Fourier transformed age-structure. The filtered version of the input time series is then obtained by taking the inverse Fourier transform, and then re-introducing the linear time trend. Monte Carlo tests of the algorithm showed that the filtering operation performed better after detrending, presumably because of the reduced influence of high frequency artifacts.

Note that the algorithm requires that both the input time series and age-structure be of odd length, and will reduce the lengths specified by one if necessary without warning to the user. It is the

user's responsibility to account for this if it might affect subsequent calculations. If necessary, the procedure FT_Coeff (Pascal version) or subroutine FTCOEF (Fortran version) may be modified by the user to handle even length inputs using Schubert's (1963) modification, but the loss of one data point in most applications is unlikely to justify the effort. Some further discussion of this point is given in Welch (1987a).

PASCAL VERSION

To improve the readability of the Pascal code each of the Procedures listed in Appendix I have been separated so that the reader may more easily examine their relationship. When implementing the code it is important that the procedures DeTrend, ReTrend, InverseFT, Attenuate, FT_Coeff, and CalculateFilterGain be inserted within Procedure Filter in the sequence indicated, so that they remain local to Procedure Filter. This has already been done for the machine-readable version of the code.

The following declarations must then be made in the calling program to specify the maximum length of the time series vector, and to declare the variables used to contain the data passed to Filter:

```
-----------------------------------------------------------------------
Program Main;

Const
  MaxLenX   =  51;       {Maximum length of time series to be analyzed}
  MaxN      =  25;          {Maximum number of Fourier Coefficients}
                                      { MaxN = MaxLenX div 2 }
Type
  TSVector   =  Array[1..MaxLenX] of Real;   {a Time Series Vector}
Var
  X          :  TSVector;       {the input time series to be filtered}
  FiltX      :  TSVector;          {the filtered time series output}
  RED        :  TSVector;     {the vector of age-structure coefficients}
                              {determining the gain of the ideal filter}
  LenX       :  Integer;    {the actual length of the input time series}
                                          { LenX <= MaxLenX }
  LenRED     :  Integer;       {the length of the age-structure used to}
                               {define the filter; LenRED <= MaxLenX}
-----------------------------------------------------------------------
```

The user must, of course, declare any other variables used in the calling program. In the first example shown below it is assumed that the variable "i" has been declared as integer.

FORTRAN VERSION

The Fortran subroutines are listed in Appendix II. They should be inserted at the end of the user's calling program in the sequence indicated. (The machine-readable version of the code is already in this order). The following declarations must then be made in the calling program to specify the maximum length of the arrays holding the time series to be filtered and the age-structure coefficients:

```
      PROGRAM MAIN
      REAL X(51),FILTX(51),RED(51)
      DOUBLE PRECISION A(26),B(26),FCFREQ(26),FLTGN(26),GNFRQ(26)
      INTEGER LENX,LENRED,MAXLEN
C DOUBLE PRECISION ARRAYS MUST HAVE MINIMUM DIMENSION EQUAL TO ONE-HALF
C LENGTH OF TIME SERIES ARRAYS X AND RED PLUS ONE; DEFINED IN MAXLEN.
      MAXLEN=26
                .
                .
                (SPECIFY LENGTH OF ARRAYS IN LENX & LENRED,
                .         AND READ DATA INTO ARRAYS X AND RED)
                .
      CALL FILTER(X,LENX,RED,LENRED,FILTX,A,B,FCFREQ,FLTGN,GNFRQ,MAXLEN)
                .
                .
                (POST-PROCESSING OF FILTERED RESULTS)
                .
                .
      STOP
      END
```

Note that in this example it is assumed the maximum length of the time series to be filtered will be ≤ 51 points, and the length of the arrays X, FILTX, and RED have been defined accordingly. The arrays A, B, FCFREQ, FLTGN, and GNFRQ have to be defined and passed to the subroutine FILTER by the user. Their dimension must also be passed, in this case specified in MAXLEN, and should be calculated as one-half the length of the longest time series array plus one; i.e. MAXLEN = INT(51/2)+1. If the user finds this definition awkward, the dimension of these double precision arrays can be (1) set equal to that of X, RED, and FILTX, and (2) LENX specified in place of MAXLEN in the argument list for FILTER above. I would in fact advocate the latter approach when feasible; my only reason for calculating the exact minimum array sizes needed during the use of FILTER in this example is for microcomputers with very limited memory.

TEST PROGRAMS

The following two programs are designed to test the correctness of the filtering code on the user's own computer using the examples discussed in Welch (1987c). The specific numerical values listed in Tables 1 & 2 were obtained using the 8087 implementation of the Turbo Pascal(C) compiler (which uses 8-byte reals), but results should be similar for other implementations. For example, running the Fortran versions of the example programs using the WATFOR-77(C) compiler gave similar but less precise results, since this version uses 4 byte real values.

The first program, Prog1, is designed to quickly establish whether the algorithm is working correctly. The input data is the sum of two sinusoids, with periods of 4 and 7 "years". In the first example using this data the age-structure defining the filtering criteria is of the form [..0,0,1,0,0..]; i.e., only one age-group contributes to forming the "stock", so the population density will show the same variability as recruitment, independent of frequency. The output from the filtering operation should be the same as the input, as is evident from the the value of the residuals shown in Table 1, and from the graph of the input and filtered output time series (Fig. 1).

The second example in Prog1 shows the effect of changing the specified age-structure to the form [..0,1,1,1,1,0,..]; i.e., one where four age-groups make equal contributions to forming the stock. Since all four age-groups contribute equally, it is obvious that the four year sinusoidal periodicity evident in the recruitment data will be completely removed in forming the stock. This is indeed the case, as the data in Table 1 and Fig. 1 demonstrate, since only the 7-yr periodicity is evident after filtering.

The second program, Prog2, provides a more biologically realistic model. An iteroparous population is defined where the adult stock is made up of the previous T=5 recruitments, and recruitment is defined as

$$y_{t+1} = C * (X_t)^{2/3} + noise_t.$$

Noise is defined as a sinusoidal fluctuation at a frequency of either s=0.2 yr^{-1} (or s=0.4 yr^{-1}), and filtering takes place according to the adult age-structure. The example demonstrates the ideal situation when filtering can identify and remove all of the noise from the recruitment data, leaving only the density-dependent stock-recruitment information. Table 2 provides the output data from the model, and Fig. 2 demonstrates the fit of the filtered recruitment data to the stock-recruitment function specified. Note that the value of C in subroutine SRR is set to less than 1 so that population size declines with time.

PROGRAM 1: PASCAL VERSION

```
Program Prog1;
{insert main declarations here}
  i   :  Integer;
{insert Procedure Filter here}
Begin
 LenX:=21; LenRED:=LenX;
     For i:=1 to LenX do begin
         X[i]:=Sin(2*Pi*i/7) + Sin(2*Pi*(i+2)/4);  RED[i]:=0 end;
 RED[1]:=1;
   Filter(X,LenX,RED,LenRED,FiltX);     {1st example; the "do-nothing" filter}
      WriteLn(CON,' Results for Example # 1');
      WriteLn(CON,' t       X(t)    FiltX(t)    Residual');
           For i:= 1 to LenX do
               WriteLn(CON,i:3,X[i]:10:6,FiltX[i]:10:6,' ',(X[i]-FiltX[i]):10);
 For i:= 1 to 4 do RED[i]:=1;
    Filter(X,LenX,RED,LenRED,FiltX);         {2nd example; the box-car filter}
       WriteLn(CON,' Results for Example # 2');
       WriteLn(CON,' t       X(t)    FiltX(t)    Residual');
            For i:= 1 to LenX do
                WriteLn(CON,i:3,X[i]:10:6,FiltX[i]:10:6,' ',(X[i]-FiltX[i]):10)
end.
```

PROGRAM 1: FORTRAN VERSION

```
      PROGRAM PROG1
      REAL X(25),FILTX(25),RED(25),PI
      DOUBLE PRECISION A(13),B(13),FCFREQ(13),FLTGN(13),GNFRQ(13)
      INTEGER LENX,LENRED,MAXLEN
C DEFINE LENGTH OF DOUBLE PRECISION ARRAYS PASSED TO SUBROUTINE.
      MAXLEN=13
C
C EXAMPLE 1.        THE "DO-NOTHING" FILTER.
      LENX=21
      LENRED=LENX
      PI=3.141592654
      DO 1 I=1,LENX
           X(I)=SIN(2.*PI*I/7.) + SIN(2.*PI*(I+2)/4.)
  1        RED(I)=0.0
      RED(1)=1.0
      CALL FILTER(X,LENX,RED,LENRED,FILTX,A,B,FCFREQ,FLTGN,GNFRQ,MAXLEN)
      PRINT*,' DO-NOTHING FILTER RESULTS.'
      DO 2 I=1,LENX
           RESID=X(I)-FILTX(I)
  2        WRITE(*,100)I,X(I),FILTX(I),RESID
C EXAMPLE 2.        THE "BOX-CAR" FILTER.
      DO 3 I=1,4
  3      RED(I)=1.
      CALL FILTER(X,LENX,RED,LENRED,FILTX,A,B,FCFREQ,FLTGN,GNFRQ,MAXLEN)
      PRINT*,' BOX-CAR FILTER RESULTS.'
      DO 4 I=1,LENX
         RESID=X(I)-FILTX(I)
  4      WRITE(*,100)I,X(I),FILTX(I),RESID
      STOP
 100  FORMAT(I3,2F15.6,'  ',E15.8)
      END
```

Table 1. Output from Pascal Program Prog1. Results for the Fortran version are similar.

Results for Example # 1

t	X(t)	FiltX(t)	Residual
1	-0.218169	-0.218169	-3.69E-015
2	0.974928	0.974928	-6.66E-016
3	1.433884	1.433884	-4.44E-016
4	-0.433884	-0.433884	7.772E-016
5	-1.974928	-1.974928	8.882E-016
6	-0.781831	-0.781831	-9.99E-016
7	1.000000	1.000000	-1.33E-015
8	0.781831	0.781831	-1.11E-016
9	-0.025072	-0.025072	3.886E-016
10	0.433884	0.433884	5.551E-016
11	0.566116	0.566116	9.992E-016
12	-0.974928	-0.974928	1.887E-015
13	-1.781831	-1.781831	-1.33E-015
14	-0.000000	0.000000	-3.19E-015
15	1.781831	1.781831	-1.78E-015
16	0.974928	0.974928	2.331E-015
17	-0.566116	-0.566116	2.220E-015
18	-0.433884	-0.433884	1.998E-015
19	0.025072	0.025072	2.331E-015
20	-0.781831	-0.781831	2.554E-015
21	-1.000000	-1.000000	-1.44E-015

Results for Example # 2

t	X(t)	FiltX(t)	Residual
1	-0.218169	0.277906	-4.96E-001
2	0.974928	0.392706	5.822E-001
3	1.433884	0.305433	1.128E+000
4	-0.433884	-0.192755	-2.41E-001
5	-1.974928	-0.600956	-1.37E+000
6	-0.781831	-0.464210	-3.18E-001
7	1.000000	0.047773	9.522E-001
8	0.781831	0.450238	3.316E-001
9	-0.025072	0.503188	-5.28E-001
10	0.433884	0.242053	1.918E-001
11	0.566116	-0.199981	7.661E-001
12	-0.974928	-0.554267	-4.21E-001
13	-1.781831	-0.485780	-1.30E+000
14	-0.000000	0.019151	-1.92E-002
15	1.781831	0.488318	1.294E+000
16	0.974928	0.513522	4.614E-001
17	-0.566116	0.179567	-7.46E-001
18	-0.433884	-0.152470	-2.81E-001
19	0.025072	-0.504059	5.291E-001
20	-0.781831	-0.740725	-4.11E-002
21	-1.000000	-0.524652	-4.75E-001

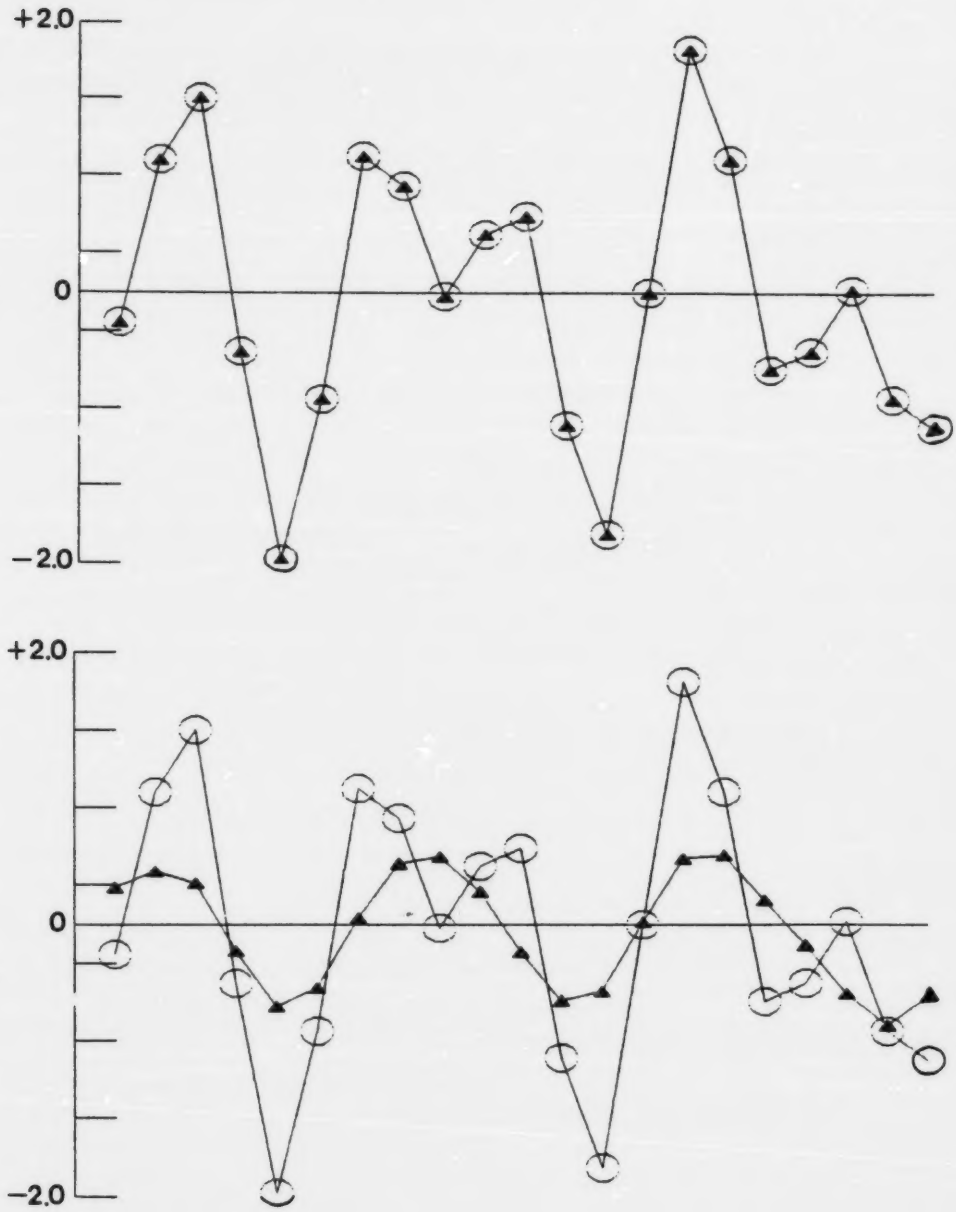

Fig. 1. Top Panel: Effect of the "do-nothing" filter on the recruitment time series.
Bottom Panel: Effect of filtering according to the [..0,1,1,1,1,0,...] age-structure. For both panels, an open circle denotes the unfiltered recruitment data, and a closed triangle indicates the same data after filtering.

PROGRAM 2: PASCAL VERSION

```
Program Prog2;
{This program is designed to test the correctness of the filtering
             algorithm on the user's computer}
Const
  MaxLenX   =  51;       {Maximum length of time series to be analyzed}
  MaxN      =  25;          {Maximum number of Fourier Coefficients}
                                { MaxN = MaxLenX div 2 }
Type
  TSVector    =  Array[1..MaxLenX] of Real;    {a Time Series Vector}
Var
  X           :  TSVector;       {the input time series to be filtered}
  FiltX       :  TSVector;          {the filtered time series output}
  RED         :  TSVector;     {the vector of age-structure coefficients}
                               {determining the gain of the ideal filter}
  LenX        :  Integer;    {the actual length of the input time series}
                                                 { LenX <= MaxLenX }
  LenRED      :  Integer;       {the length of the age-structure used to}
                                 {define the filter; LenRED <= MaxLenX}
Const   T           =  5;
Var     Stock       :  TSVector;
        i           :  Integer;
Procedure SRR(yr:integer; var stock,recruits: Real);
Const A=0.40; C=0.75; S=0.2; Var j:integer;
Begin stock:=0;
      For j:=0 to T-1 do stock:=stock+RED[j+1]*X[yr-T+j];
      Recruits:=C*Exp((2/3)*Ln(Stock)) + A*Sin(2*Pi*S*yr);
end; {Procedure SRR}

{INSERT PROCEDURE Filter HERE}

Begin LenX:=51-T; LenRED:=LenX;
    For i:=1 to LenX do RED[i]:=0;
    For i:=1 to T do RED[i]:=1/T;              {Specify rectangular age-structure}
{Start simulation}
    For i:=1 to T do begin X[i]:=1; Stock[i]:=1 end;       {initial equilibrium}

    For i:=T+1 to T+LenX do SRR(i,stock[i],X[i]);

    For i:=1 to LenX do begin stock[i]:=stock[i+T]; X[i]:=X[i+T] end;
    Filter(X,LenX,RED,LenRED,FiltX);
       WriteLn(LST'   t  stock(t)   recr(t)  F_recr(t)');
    For i:=1 to LenX do Writeln(LST,i:4,stock[i]:10:6,X[i]:10:6,FiltX[i]:10:6);
end.
```

PROGRAM 2: FORTRAN VERSION

```
      PROGRAM PROG2
      REAL X(50),FILTX(50),RED(45),PI,STOCK(50)
      DOUBLE PRECISION A(26),B(26),FCFREQ(26),FLTGN(26),GNFRQ(26)
      INTEGER LENX,LENRED,MAXLEN,T
C FIRST DEFINE LENGTH OF DOUBLE PRECISION ARRAYS PASSED TO SUBROUTINE.
      MAXLEN=26
      LENX=45
      LENRED=LENX
      PI=3.141592654
      T=5
      DO 1 I=1,LENX
    1 RED(I)=0.
      DO 2 I=1,T
    2 RED(I)=1./T
C START SIMULATION
      DO 3 I=1,T
      X(I)=1
    3 STOCK(I)=1
      I1=T+1
      I2=T+LENX
      DO 4 I=I1,I2
    4 CALL SRR(I,T,X,LENX,RED,LENRED,STOCK(I),X(I))
      DO 5 I=1,LENX
      I1=I+T
      STOCK(I)=STOCK(I1)
    5 X(I)=X(I1)
      CALL FILTER(X,LENX,RED,LENRED,FILTX,A,B,FCFREQ,FLTGN,GNFRQ,MAXLEN)
      PRINT*,'  T  STOCK(T)   RECR(T)  F_RECR(T)'
      DO 6 I=1,LENX
    6 WRITE(*,100)I,STOCK(I),X(I),FILTX(I)
      STOP
  100 FORMAT(I4,3F10.6)
      END
CCCC
```

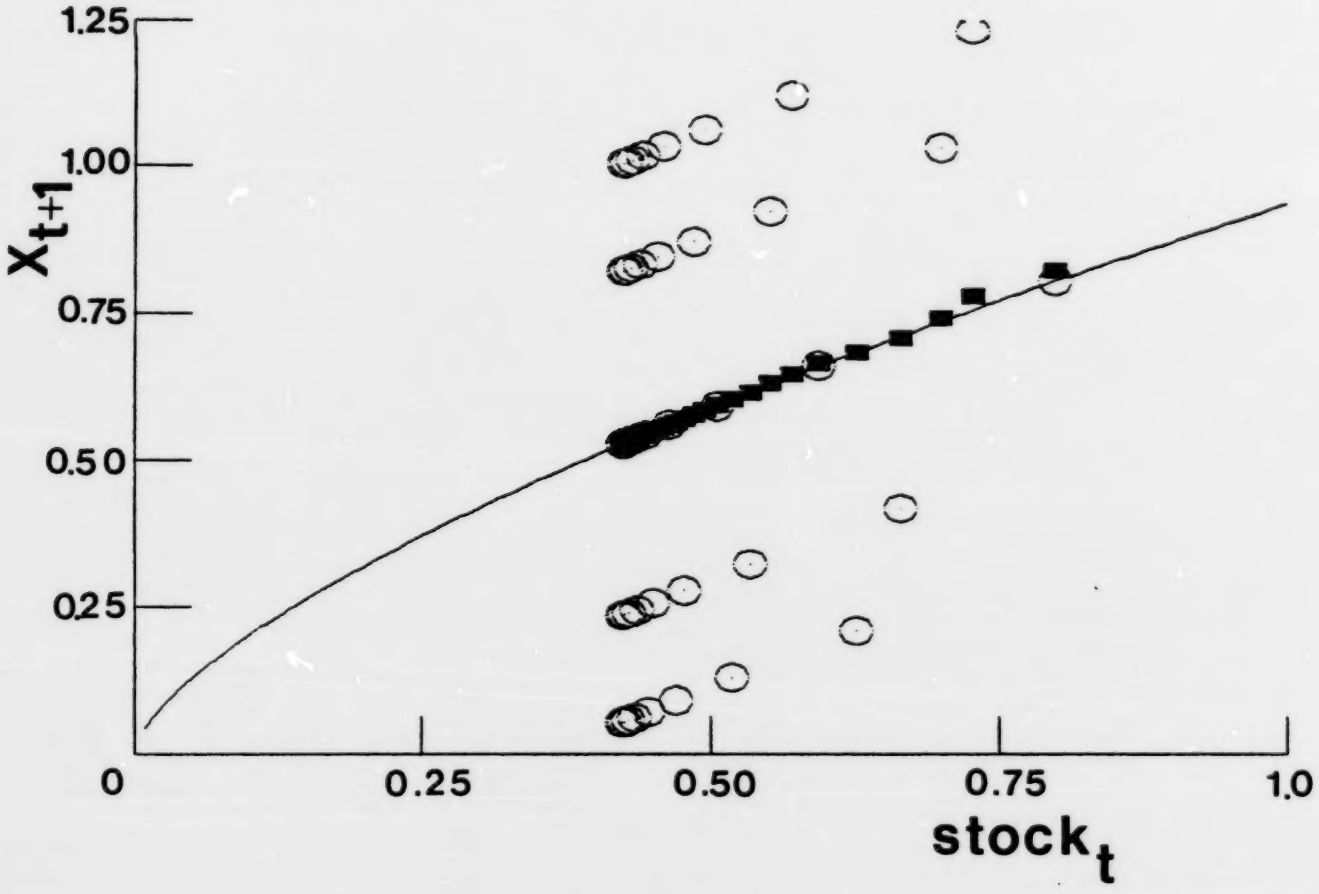

Fig. 2. An example where filtering can remove all of the density-independent noise. The stock-recruitment function for the model is also shown. Open circle: unfiltered recruitment data; closed triangle: filtered data.

Table 2. Output from Pascal Program Prog2. Results for the Fortran version are similar.

t	stock(t)	recr(t)	F_recr(t)	t	stock(t)	recr(t)	F_recr(t)
1	1.000000	1.130423	0.740605	23	0.449866	0.205221	0.439122
2	1.026085	0.998100	0.741450	24	0.446368	0.057627	0.437122
3	1.025705	0.527684	0.730182	25	0.443323	0.436055	0.435492
4	0.931241	0.334792	0.701843	26	0.440666	0.814734	0.433628
5	0.798200	0.645360	0.663136	27	0.438341	0.667896	0.431930
6	0.727272	0.986962	0.625258	28	0.436307	0.196328	0.430591
7	0.698580	0.825594	0.594733	29	0.434528	0.049846	0.429763
8	0.664078	0.335761	0.568531	30	0.432972	0.429241	0.429135
9	0.625694	0.168236	0.548262	31	0.431609	0.808762	0.428080
10	0.592383	0.529009	0.532608	32	0.430415	0.662663	0.427129
11	0.569112	0.895485	0.517719	33	0.429368	0.191742	0.426489
12	0.550817	0.739078	0.503890	34	0.428451	0.045825	0.426355
13	0.533514	0.258240	0.492197	35	0.427647	0.425714	0.426324
14	0.518010	0.103326	0.483133	36	0.426941	0.805668	0.425474
15	0.505028	0.475632	0.475774	37	0.426322	0.659949	0.424811
16	0.494352	0.849328	0.468589	38	0.425780	0.189360	0.424615
17	0.485121	0.698164	0.462018	39	0.425303	0.043735	0.425203
18	0.476938	0.222714	0.456508	40	0.424885	0.423879	0.425702
19	0.469833	0.072847	0.452231	41	0.424518	0.804058	0.423529
20	0.463737	0.449341	0.448694	42	0.424196	0.658535	0.423306
21	0.458479	0.826360	0.445097	43	0.423913	0.188119	0.425972
22	0.453885	0.678068	0.441826	44	0.423665	0.042645	0.433407
				45	0.423447	0.422923	0.443568

SUMMARY

The preceding sections have described the use and testing of the filtering codes on the reader's computer. I trust that you will find their use rewarding, and that the process of using them on your own favourite data set will set you thinking about the nature of stock recruitment analysis with more care. My experience has been, however, that most biologists have conceptual difficulty thinking in terms of a set of frequencies rather than with the equivalent time series, and are understandably reluctant to use a tool whose function they only roughly grasp. It is all too easy to misinterpret explanations and use an unfamiliar tool in an inappropriate manner.

This section is therefore intended as a step-by-step guide to getting your analysis off the ground. First, start by thinking about the nature of your data, and not by wondering about how much recruitment variability the filter will suppress. In particular, think about the appropriate definition of stock. It may be that only one time series is available that can be used as a proxy for "stock". If this is the case then you will have to hope that your proxy will show reasonable coherence with better measures of stock over time.

Routine sampling of commercial fisheries often permits a much wider range of candidate time series to be considered, however. Think carefully. Are the adults (or juveniles) highly piscivorous, so that cannibalism may play a role in population regulation? Or is it more likely that regulation is achieved through a prerecruit mortality rate dependent on the abundance of the 0-group? In these cases stock should then be measured as the sum of the number of individuals per age-class, weighted by either relative rates of (i) predation or (ii) fecundity. Other situations may suggest themselves to the investigator, although these two are surely the most obvious.

Second, now that you have a candidate time series to compare with recruitment, think about how the stock time series was formed from <u>past</u> recruitments. I find it often helps to write down the process and to separate the components. For example, assume recruits are defined as one year old individuals and stock is defined as the population fecundity (total egg production). Then the contribution of recruits from x years ago to this year's population fecundity is then calculated as the fraction surviving over x years times the fraction sexually mature at age x+1 times the fecundity (eggs per mature individual) at age x+1, or

$$k_{x+1} = \text{maturity}_{x+1} * \text{fecundity}_{x+1} * \exp(-Z*x).$$

For simplicity I have assumed that the instantaneous total mortality rate is the same for all ages; in practice the last term may be better replaced by the sum of the estimated age-specific rates, i.e. $\sum_{i=1}^{i=x} Z_i$. If possible, all rates should also be for <u>females</u> if such sex-specific information exists.

By doing this for each age group the sequence of k_a values which describe how past recruits form the stock is generated. The components of this process should have been reasonably time-invariant over the period for which the stock and recruitment data are available, but my experience has been that small variations in the value of the k_a coefficients (i.e. on the order of the sampling uncertainty) do not have a major influence on the results. This assumption can, however, be easily checked by doing the filtering analysis again with a slightly different set of age structure coefficients, and comparing the results.

What remains is to filter the data! At this point all one needs to do is to invoke the filtering procedure, passing the recruitment data and the age-structure coefficients. The procedure then returns the appropriately filtered data for the user's further analysis. Remember, the key to this process is to specify the age-structure coefficients describing how stock is determined by past recruitment levels. The filter then removes that fraction of the recruitment variability which cannot be associated with variation in the measure of stock under consideration.

One final point is worth emphasizing. Properly used, filtering is a valuable tool in clarifying what sort of functional relationship might describe the stock-recruitment curve, and in increasing the precision of the parameter estimates from a fitted curve (Welch 1987a). It is possible to misuse the technique, however, by deliberately choosing a measure of stock with very large age-structure, so that a lot of filtering takes place before curve fitting. It is important that filtering is done after some careful thinking is done about what type of processes may be occurring, and to use filtering within the confines of reason. Good luck! I welcome both comments and criticisms on the technique, and I trust that the reader will find the use of the technique rewarding.

ACKNOWLEDGMENTS

I thank Drs. Tim Mulligan and Don Noakes for their effort in reviewing the manuscript.

LITERATURE CITED

Evans, G. 1987 Filtering recruitment time series: Comment.
Can. J. Fish. Aquat. Sci. 44(3): 691-692

Goertzel, G. 1960 Fourier analysis. Chapter 24, p. 258-262,
in: Ralston, A. and H.S. Wilf [eds.]
Mathematical methods for digital computers. John Wiley & Sons Ltd.

Schubert, G.R. 1963 Remark on algorithm 157, Fourier series
approximation. Reprinted in: Collected algorithms from ACM
(1980), Vol. 1, 157-P 1-0.

Welch, D.W. 1986 Identifying the stock-recruitment relationship
for age-structured populations using time-invariant matched
linear filters. Can. J. Fish. Aquat. Sci. 43(1): 108-123.

Welch, D.W. 1987a Frequency domain filtering of age-structured
population data. Can. J. Fish. Aquat. Sci. 44(3): 605-618

Welch, D.W. 1987b Use and precision of age-structure based
recruitment filtering theory: reply to Evans.
Can. J. Fish. Aquat. Sci. 44(3): 692-694

Welch, D.W. 1987c Influence of three life history strategies on the
information content of stock-recruitment data.
Can. J. Fish. Aquat. Sci. 44(7): [in press, July].

Appendix I. Frequency domain filtering algorithms: Pascal code.

```
Procedure Filter(X : TSVector; var LenX : Integer; RED : TSVector;
                      LenRED : Integer; var FiltX : TSVector);
{                                          written : 13 May 1985
                                       Last Modified : 24 May 1985
    This program does the bookkeeping for filtering an input time series.
    It makes the appropriate procedure calls to define the spectral shape
    used in attenuating the variance of an input time series, and then returns
    the filtered time series and its length. Note that the name of the
    parameter RED reflects the assumption that the spectral shape
    of the ideal gain will be determined from the distribution of reproductive
    effort; however, in some cases the input impulse response might more
    appropriately be some other function-- for example, the stable age
    distribution, which is determined solely by the survivorship schedule.
}
Type
    FTCoeffVector  =  Array[1..MaxN] of Real;
Var
   A,B,FourierFreq :  FTCoeffVector; {Fourier transform coefficients
                                      & frequencies at which the
                                       recruitment time series is evaluated}
   Gain,GainFreq   :  FTCoeffVector; {Gain of RED & Fourier frequencies
                                          at which it is evaluated}
   NFreqs,NGains   :  Integer;       {Number of Fourier Frequencies at
                                        which the recruitment and RED
                                          transforms are evaluated}

  Slope, Intercept :  Real;          {Define the linear time trend in the
                                        input time series}


{INSERT PROCEDURES CalculateFilterGain, DeTrend, FT_Coeff, Attenuate,
                   InversFT, and ReTrend HERE}


begin {Procedure Filter}
    CalculateFilterGain(RED,LenRED,Gain,GainFreq,NGains); {compute ideal gain}
    DeTrend(X,LenX,Slope,Intercept);              {remove the linear time trend}
    FT_Coeff(X,LenX,A,B,FourierFreq,NFreqs);   {calculate Fourier coefficients}
    Attenuate(A,B,FourierFreq,NFreqs,Gain,GainFreq,NGains); {modify the coeffs}
    InverseFT(A,B,FourierFreq,NFreqs,FiltX);
    ReTrend(Slope,Intercept,FiltX,LenX);       {add the linear time trend}
end; {of procedure Filter}
```

(PASCAL CODE CONT'D...)

```
{=============================================================================}
Procedure DeTrend(var X : TSVector; LenX : Integer;
                  var Slope, Intercept : Real);
{                                       written : 24 May 1985
                                   last modified : 24 May 1985
 This procedure subtracts the linear time trend from a time series prior
 to filtering.
}
var
  SumXY,SumXsq,SumX,SumY    :   Real;
  i                         :   Integer;
Begin
     SumXY := 0.0;  SumXsq := 0.0; SumX := 0.0; SumY := 0.0;
     For i := 1 To LenX do begin    (note that the independent variable is)
         SumXY  := SumXY + X[i]*i;  ( i & the dependent variable is X[i] )
         SumXsq := SumXsq + i*i;
         SumX   := SumX + i;
         SumY   := SumY + X[i] end;
  Slope     := (SumXY - (SumX * SumY)/LenX) / (SumXsq - Sqr(SumX)/LenX);
  Intercept := (SumY - Slope * SumX)/LenX;
    For i := 1 To LenX do
        X[i] := X[i] - Intercept - Slope * i;
end; (of procedure DeTrend)
```

```
{=============================================================================}
Procedure ReTrend(Slope,Intercept: Real; var FiltX: TSVector; LenX: Integer);
{                                    written : 24    May 1985
                                last modified : 21 August 1985
 This procedure adds the linear trend back to a detrended & filtered TS.
 }
 var
    i    : Integer;
Begin
     For i := 1 to LenX do
         FiltX[i] := FiltX[i] + Intercept + Slope * i
end; (of procedure ReTrend)
```

(PASCAL CODE CONT'D...)

```
{=============================================================================}
Procedure InverseFT(A,B,Freq : FTCoeffVector;  NFreqs : Integer;
                                              var NewTS : TSVector);
{                                          written :   2    May 1985
                                        last modified :  20 August 1985
   - This procedure sums a vector of Fourier coefficients A,B, of length
     Nfreqs to obtain the TS representation, based on the forward
     summation of the Fourier series.
}
var
   i,j            : Integer;
Begin
     For i := 1 To 2 * NFreqs - 1 Do
       Begin
         NewTS[i] := A[1];
            For j := 2 To NFreqs Do
                NewTS[i] := NewTS[i] +
                     2*(A[j]*Cos((i-1) * Freq[j]) + B[j]*Sin((i-1) * Freq[j]));
       end;
end; { of procedure InverseFT}



{=============================================================================}
Procedure CalculateFilterGain(ImpulseResponse : TSVector;
          LengthOfImpulseResponse : Integer; var Gain,GainFreq : FTCoeffVector;
          var NGains : Integer);
{                                       written : 15 May 1985
                                     last modified : 21 May 1985
 This procedure is used to calculate the gain for an input age-structure
(= ImpulseResponse) and then returns the gain and the frequencies at which
 it was calculated.
 }
var   i       :  Integer;
      C,D     :  FTCoeffVector;  {Local arrays holding Fourier coefficients
                                  of the impulse response}
SumImpulseResponse
              : Real;
begin
          SumImpulseResponse := 0.0;
             For I := 1 to LengthOfImpulseResponse do
                 SumImpulseResponse := SumImpulseResponse + ImpulseResponse[i];

             SumImpulseResponse := SumImpulseResponse/LengthOfImpulseResponse;
                                    (Normalize so that the gain of
                                          the filter is 1.0 at 0.0 frequency)
             For i := 1 to LengthOfImpulseResponse do
                 ImpulseResponse[i] := ImpulseResponse[i]/SumImpulseResponse;

     FT_Coeff(ImpulseResponse,LengthOfImpulseResponse,C,D,GainFreq,NGains);
     For i := 1 to NGains do
         Gain[i] := Sqrt( Sqr(C[i]) + Sqr(D[i]));
end; {of procedure CalculateFilterGain}
```

(PASCAL CODE CONT'D...)

```
{=============================================================================}
Procedure Attenuate(var A,B,FourierFreq : FTCoeffVector; Nfreqs : Integer;
                    Gain,GainFreq :FTCoeffVector; NGains : Integer);
(*
                                  written :  6 May 1985
                              last modified : 21 May 1985
   This procedure multiplies the set of Fourier coefficients (A), (B) by a
   predetermined set of frequency-specific constants (the "gain"). This
   attenuates the variance without distorting the phase information. To
   obtain the original TS it is necessary to calculate the inverse transform
   of the modified Fourier coefficients.

   The gain at the Fourier frequencies is obtained from
   (i)  linear interpolation between adjacent frequencies at which the input
        gain was specified (if GainFreq <> FourierFreq), or
   (ii) linear extrapolation if FourierFreq > GainFreq.
*)
var     InterpolGain      :    Real;     {The linearly interpolated gain}
        i,j               :    Integer;

Begin
  For i := 1 to NFreqs do begin
    j := 1;
        While j < NGains do begin
            If ((FourierFreq[i] >= GainFreq[j]) and
                     (FourierFreq[i] < GainFreq[j+1])) then begin
              InterpolGain := Gain[j] + (Gain[j+1] - Gain[j])*
                (FourierFreq[i] - GainFreq[j]) / (GainFreq[j+1] - GainFreq[j]);
              A[i] := InterpolGain * A[i];
              B[i] := InterpolGain * B[i];
              j := NGains end
            Else j := j + 1
        end; {while}
            If FourierFreq[i] >= GainFreq[NGains] Then begin
                  InterpolGain := Gain[NGains] +
                                  (Gain[Ngains] - Gain[NGains-1]) *
                                  (FourierFreq[i] - GainFreq[NGains]) /
                                  (GainFreq[NGains] - GainFreq[NGains-1]);
                  A[i] := InterpolGain * A[i];
                  B[i] := InterpolGain * B[i]
            end;
  end;
end;   {of procedure Attenuate}
```

(PASCAL CODE CONT'D...)

```
{=============================================================================}
Procedure FT_Coeff(X : TSVector; var LenX : Integer;
                   var A,B,Freq : FTCoeffVector; var NFreqs : Integer);
(*                                 Written: 30 April  1985
                                Last Modified: 19 August 1985

  - This procedure calculates the Fourier coefficients of a time series X of
    length LenX.

  - If LenX is even, then the last value of the time series
    is excluded from the calculation of the FT coefficients. For recruitment
    time series this corresponds to the most recent recruitment estimate,
    which is most poorly estimated by VPA techniques.

  - Reference:  Goertzel, G. (1960) Fourier Analysis. Chapter 24,
                p. 258-262 in: Ralston, A. and H.S. Wilf [Eds.]
                Mathematical Methods for Digital Computers.
                John Wiley and Sons Ltd.
*)
var
   N     :  Integer;    {Index used in computations; for computational
                         reasons the TS must be of odd length, so
                          N is defined below as (LenX-1)Div 2}
   I,J   :  Integer;                {Dummy indices}

   C,C1,Coef,Constant,Q,S,S1,U0,U1,U2,XZero
         :  Real;      {Constants used in the estimation procedure}

Begin
          N := (LenX-1) Div 2;
          If(LenX mod 2 = 0) Then LenX := LenX - 1;
          C := 1.0;  S := 0.0;                        {Specify starting values}
          Coef := 1.0/(2.0*N + 1.0);      {The specification of these 2 values}
          Constant := 2.0 * Pi*Coef;       {differs from the above references}
          C1 := Cos(Constant);   S1 := Sin(Constant);
          XZero := X[1];
          NFreqs := N+1;

  For J := 1 to NFreqs Do
    Begin                                          {Recursive Calculation of}
          U1 := 0.0; U2 := 0.0;                      {Fourier Coefficients}
            For i := 2*N + 1 DownTo 2 Do
              Begin
                   U0 := X[i] + 2.0*C*U1 - U2;
                   U2 := U1;
                   U1 := U0
              end;
          A[J]      := Coef * (XZero + C * U1 - U2);
          B[J]      := Coef * S * U1;
          Freq[J]   := Constant * (J-1);
          Q         := C1 * C  -  S1 * S;
          S         := C1 * S  +  S1 * C;
          C         := Q
    end;    {of Do}
end;   {of procedure FT_Coeff}                    (PASCAL CODE COMPLETED)
```

Appendix II. Frequency domain filtering algorithms: Fortran code.

```
      SUBROUTINE FILTER(X,LENX,RED,LENRED,FILTX,A,B,FCFREQ,FLTGN,GNFRQ,
     *                  MAXLEN)
      REAL X(LENX),RED(LENRED),FILTX(LENX),SLOPE,INTCPT
      DOUBLE PRECISION A(MAXLEN),B(MAXLEN),FCFREQ(MAXLEN),
     *                 FLTGN(MAXLEN),GNFRQ(MAXLEN)
CCC
C  -NOTE THAT IF THE LENGTHS OF THE INPUT TIME SERIES X OR RED ARE EVEN
C       THEIR RESPECTIVE VALUES ARE REDUCED BY 1 BEFORE RETURNING TO THE
C       CALLING PROGRAM. IT IS THE RESPONSIBILITY OF THE USER TO ACCOUNT
C       FOR THIS IF THE LENGTH OF THE AVAILABLE TIME SERIES IS IMPORTANT
C       FOR LATER CALCULATIONS.
CCC
C       A,B       -THE FOURIER COEFFICIENTS OF THE FOURIER TRANSFORMED
C                      TIME SERIES. CALCULATED IN DOUBLE PRECISION.
C       FCFREQ    -THE FREQUENCIES AT WHICH THE FOURIER COEFFICIENTS OF THE
C                      INPUT TIME SERIES ARE CALCULATED
C       FILTX     -THE FREQUENCY DOMAIN FILTERED TIME SERIES
C       FLTGN     -THE CALCULATED GAIN OF THE IDEAL FILTER
C       GNFRQ     -THE FREQUENCIES AT WHICH THE IDEAL GAIN IS CALCULATED
C       INTCPT    -THE INTERCEPT OF THE LINEAR TREND IN THE INPUT TIME SERIES
C       LENFC     -NUMBER OF FOURIER COEFFICIENT PAIRS (A,B) CALCULATED
C                      FROM A TIME SERIES OF LENGTH LENX
C       LENRED    -THE LENGTH OF THE SPECIFIED AGE-STRUCTURE
C       LENX      -LENGTH OF INPUT TIME SERIES; MAY BE CHANGED BY SUBROUTINE!!
C       MAXLEN    -MAXIMUM DIMENSION OF INPUT ARRAYS; USED AS DUMMY ARGUMENT
C                      FOR SUBROUTINE ARRAY SIZES.
C       NFREQS    -THE NUMBER OF FREQUENCIES AT WHICH THE FOURIER COEFFICIENTS
C                      OF THE INPUT TIME SERIES ARE CALCULATED
C       NGAINS    -THE NUMBER OF FREQUENCY-SPECIFIC GAINS CALCULATED
C       RED       -THE SET OF AGE-STRUCTURE COEFFICIENTS DEFINING HOW "STOCK"
C                      IS MADE UP OF PAST RECRUITING AGE-CLASSES
C       SLOPE     -THE SLOPE OF THE LINEAR TREND IN THE INPUT TIME SERIES
C       X         -THE INPUT TIME SERIES.
C
C  COMPUTE IDEAL GAIN FROM AGE-STRUCTURE.
       LENFC = INT(FLOAT(LENRED)/2.0) + 1
         CALL CALCGN(RED,LENRED,A,B,FLTGN,GNFRQ,NGAINS,LENFC,MAXLEN)
C  REMOVE THE LINEAR TREND.
      CALL DETRND(X,LENX,SLOPE,INTCPT)
C  CALCULATE FOURIER COEFFICIENTS OF TIME SERIES.
      LENFC = INT(FLOAT(LENX)/2.0) + 1
        CALL FTCOEF(X,LENX,LENFC,A,B,FCFREQ,NFREQS,MAXLEN)
C  FILTER IDENTIFIABLE NOISE.
      CALL ATNUAT(A,B,FCFREQ,NFREQS,FLTGN,GNFRQ,NGAINS,MAXLEN)
C  RETRANSFORM TO TIME DOMAIN.
      CALL INVERS(A,B,FCFREQ,NFREQS,FILTX,LENX,MAXLEN)
C  RE-INTRODUCE LINEAR TREND.
      CALL RETRND(X,FILTX,LENX,SLOPE,INTCPT)
      RETURN
      END
```

(FORTRAN CODE CONT'D...)

```
      SUBROUTINE CALCGN(RED,LENRED,A,B,FLTGN,GNFRQ,NGAINS,LENFC,MAXLEN)
      REAL RED(LENRED)
      DOUBLE PRECISION A(MAXLEN),B(MAXLEN),FLTGN(MAXLEN),GNFRQ(MAXLEN)
CCC
C - THIS SUBROUTINE CALCULATES THE GAIN FOR THE AGE-STRUCTURE COEFFICIENTS
C   DESCRIBING THE CONTRIBUTION OF PAST RECRUITMENTS TO STOCK.
C
C  FIRST ENSURE THAT THE AGE-STRUCTURE COEFFICIENTS ARE NORMALIZED TO HAVE
C  AN INTEGRAL OF 1.0. THIS ENSURES THAT THE GAIN OF THE FILTER IS 1.0 AT
C  ZERO FREQUENCY. SEE WELCH(1986) CAN.J.FISH.AQUAT.SCI. 43(1):112-113
C  FOR REASONS.
C
      SUMRED = 0.0
      DO 1 I = 1,LENRED
    1 SUMRED = SUMRED + RED(I)
C
C NORMALIZE SUM OF AGE-STRUCTURE COEFFICIENTS.
C
      SUMRED=SUMRED/FLOAT(LENRED)
      DO 2 I = 1,LENRED
    2 RED(I) = RED(I)/SUMRED
C
C  CALCULATE THE FOURIER COEFFICIENTS OF THE AGE-STRUCTURE
C
      CALL FTCOEF(RED,LENRED,LENFC,A,B,GNFRQ,NGAINS,MAXLEN)
C
C  CALCULATE THE FILTER'S GAIN AT EACH FOURIER FREQUENCY.
C
      DO 3 I = 1,NGAINS
    3 FLTGN(I) = DSQRT(A(I)*A(I) + B(I)*B(I))
      RETURN
      END
```

(FORTRAN CODE CONT'D...)

```
      SUBROUTINE DETRND(X,LENX,SLOPE,INTCPT)
      REAL X(LENX),SUMXY,SUMXSQ,SUMX,SUMY,INTCPT
C
C  THIS SUBROUTINE SUBTRACTS THE LINEAR TREND FROM AN INPUT TIME SERIES
C
      SUMXY  = 0.0
      SUMXSQ = 0.0
      SUMX   = 0.0
      SUMY   = 0.0
C
C  CALCULATE THE LINEAR TREND. NOTE THAT THE INDEPENDENT VARIABLE IS
C            TIME (I) AND THE DEPENDENT VARIABLE IS X.
C
      DO 1 I = 1,LENX
      SUMXY  =  SUMXY  +  X(I)*FLOAT(I)
      SUMXSQ =  SUMXSQ +  FLOAT(I*I)
      SUMX   =  SUMX   +  FLOAT(I)
    1 SUMY   =  SUMY   +  X(I)
C
      SLOPE = (SUMXY - SUMX*SUMY/FLOAT(LENX)) / (SUMXSQ-SUMX*SUMX/LENX)
      INTCPT   = (SUMY - SLOPE*SUMX)/LENX
C
      DO 2 I = 1,LENX
    2 X(I)  =  X(I) - INTCPT - SLOPE*FLOAT(I)
      RETURN
      END
```

```
      SUBROUTINE RETRND(X,FILTX,LENX,SLOPE,INTCPT)
      REAL X(LENX),FILTX(LENX),SLOPE,INTCPT
C
C -THIS SUBROUTINE ADDS THE LINEAR TREND BACK TO THE DETRENDED INPUT AND
C       FILTERED TIME SERIES
C
      DO 1 I=1,LENX
         X(I)     = X(I)     + INTCPT + SLOPE*FLOAT(I)
    1    FILTX(I) = FILTX(I) + INTCPT + SLOPE*FLOAT(I)
      RETURN
      END
```

(FORTRAN CODE CONT'D...)

```
      SUBROUTINE FTCOEF(X,LENX,LENFC,A,B,FCFREQ,NFREQS,MAXLEN)
      REAL X(LENX)
      DOUBLE PRECISION COEF,CONST,C,S,Q,C1,S1,U0,U1,U2
      DOUBLE PRECISION A(MAXLEN),B(MAXLEN),FCFREQ(MAXLEN)
C
C -THIS SUBROUTINE CALCULATES THE FOURIER COEFFICIENTS OF A TIME SERIES
C     X OF LENGTH LENX.
C -IF LENX IS EVEN THEN THE VALUE OF LENX IS REDUCED BY 1 AND THE LAST
C     VALUE OF THE TIME SERIES IS EXCLUDED FROM CALCULATION OF THE
C     FOURIER TRANSFORM COEFFICIENTS.
C -FOR RECRUITMENT TIME SERIES THIS CORRESPONDS TO THE MOST RECENT
C     RECRUITMENT ESTIMATE, WHICH IS MOST POORLY ESTIMATED BY VPA
C     TECHNIQUES.
C
C -REFERENCE:
C             GOERTZEL,G. (1960) FOURIER ANALYSIS. CHAPTER 24 IN:
C             RALSTON,A. AND H.S. WILF (EDS.)
C             MATHEMATICAL METHODS FOR DIGITAL COMPUTERS, P.258-262
C             JOHN WILEY AND SONS LTD.
C
      N = INT( FLOAT(LENX) / 2.0)
      IF(MOD(LENX,2).EQ.0) LENX = LENX-1
      C = 1.0
      S = 0.0
C
      COEF  = 1.0 / FLOAT(LENX)
      CONST = 2.0 * 3.141592654 * COEF
C
      C1 = DCOS(CONST)
      S1 = DSIN(CONST)
      XZERO = X(1)
      NFREQS = N + 1
C
C CALCULATE THE FOURIER COEFFICIENTS RECURSIVELY.
C
      DO 2 J = 1,NFREQS
         U1 = 0.0
         U2 = 0.0
         MAX = 2*N + 1
           DO 1 I = MAX,2,-1
              U0 = X(I) + 2.0*C*U1 - U2
              U2 = U1
  1           U1 = U0
        A(J) = COEF*(XZERO + C*U1 -U2)
        B(J) = COEF * S * U1
        FCFREQ(J) = CONST * (J-1)
        Q  =  C1 * C  -  S1 * S
        S  =  C1 * S  +  S1 * C
  2     C  =  Q
      RETURN
      END
```

(FORTRAN CODE CONT'D...)

```
      SUBROUTINE ATNUAT(A,B,FCFREQ,NFREQS,FLTGN,GNFRQ,NGAINS,MAXLEN)
      DOUBLE PRECISION INTGN
      DOUBLE PRECISION A(MAXLEN),B(MAXLEN),FCFREQ(MAXLEN),FLTGN(MAXLEN),
     *                 GNFRQ(MAXLEN)
C
C -THIS SUBROUTINE MULTIPLIES THE SET OF FOURIER COEFFICIENTS (A), (B),
C       BY A PREDETERMINED SET OF FREQUENCY-SPECIFIC CONSTANTS (THE FILTER
C       GAIN).
C
C -THE GAIN AT THE FOURIER FREQUENCIES (FCFREQ) IS CALCULATED BY
C
C           (A) LINEAR INTERPOLATION BETWEEN ADJACENT FREQUENCIES AT WHICH
C               THE FILTER'S GAIN WAS SPECIFIED (IF GNFRQ .NE. FCFREQ)
C
C           (B) LINEAR EXTRAPOLATION IF FCFREQ > GNFRQ.
C
C
      DO 4 I = 1,NFREQS
         J=1
  1      IF (J.GT.NGAINS-1) GOTO 3
         JJ=J+1
           IF ((FCFREQ(I) .GE. GNFRQ(J)) .AND.
     *         (FCFREQ(I) .LT. GNFRQ(JJ))) GOTO 2
           J=J+1
           GOTO 1
C
C  CALCULATE THE FILTER'S GAIN BY INTERPOLATING BETWEEN THE GAINS CALCULATED
C            AT THE FOURIER FREQUENCIES FOR THE INPUT AGE-STRUCTURE
C
  2        INTGN=FLTGN(J)+((FLTGN(JJ)-FLTGN(J))*
     *               (FCFREQ(I) -GNFRQ(J))/(GNFRQ(JJ)-GNFRQ(J)))
           A(I) = INTGN * A(I)
           B(I) = INTGN * B(I)
           GOTO 4
C
C  LINEARLY EXTRAPOLATE TO ESTIMATE THE FILTER'S GAIN AT FREQUENCIES GREATER
C           THAN THOSE USED TO ORIGINALLY CALCULATE THE GAIN.
C
  3       IF (FCFREQ(I) .LT. GNFRQ(NGAINS)) GOTO 4
             JJ = NGAINS -1
             INTGN =  FLTGN(NGAINS) +
     *                 (FLTGN(NGAINS) - FLTGN(JJ)) *
     *                 (FCFREQ(I) - GNFRQ(NGAINS)) /
     *                 (GNFRQ(NGAINS) - GNFRQ(JJ))
             A(I) = INTGN * A(I)
             B(I) = INTGN * B(I)
   4  CONTINUE
C     PRINT*
C     DO 11 I = 1,NFREQS
C 11  WRITE(*,101)I,NFREQS,NGAINS,A(I),B(I),FLTGN(I),FCFREQ(I),GNFRQ(I)
      RETURN
  101 FORMAT(I2,2I4,5F12.8)
      END
```

(FORTRAN CODE CONT'D...)

```
      SUBROUTINE INVERS(A,B,FCFREQ,NFREQS,FILTX,LENX,MAXLEN)
      REAL FILTX(LENX)
      DOUBLE PRECISION A(MAXLEN),B(MAXLEN),FCFREQ(MAXLEN)
C
C -THIS SUBROUTINE RETRANSFORMS THE INPUT TIME SERIES TO OBTAIN THE
C       FILTERED TIME SERIES BY SUMMING THE VECTOR OF ATTENUATED FOURIER
C       COEFFICIENTS.
C
      K = 2 * NFREQS - 1
      DO 1 I = 1,K
         FILTX(I) = A(1)
           DO 1 J = 2,NFREQS
              FILTX(I) = FILTX(I) +
     *                   2.0 * ( A(J) * COS((I-1) * FCFREQ(J)) +
     *                           B(J) * SIN((I-1) * FCFREQ(J)) )
  1   CONTINUE
      RETURN
      END
```

(FORTRAN CODE COMPLETED)